NAME : KUNDAN KUMAR
ENTRY NO : 2023JCS2560

ADVANCE COMPUTER NETWORKS REPORT ASSIGNMENT 3

PART 1 (IMPLEMENTATION OF LEARNING SWITCH AND CONTROLLER HUB)

1) **In both scenarios, conducting 3 pings for each case. Report the latency values. Explain the observed latency differences between the Hub Controller and Learning Switch. Also, explain differences (if any) observed between h2 and h5 for both controller types.**

Controller Hub Data

THREE PINGS BETWEEN H2 AND H5

```
mininet> h2 ping h5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=28.9 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=7.81 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=9.02 ms
64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=7.15 ms
64 bytes from 10.0.0.5: icmp_seq=5 ttl=64 time=9.15 ms
64 bytes from 10.0.0.5: icmp_seq=6 ttl=64 time=6.38 ms
64 bytes from 10.0.0.5: icmp_seq=7 ttl=64 time=5.96 ms
64 bytes from 10.0.0.5: icmp_seq=8 ttl=64 time=10.7 ms


mininet> h2 ping h5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=8.62 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=6.31 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=8.13 ms
64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=7.82 ms
64 bytes from 10.0.0.5: icmp_seq=5 ttl=64 time=7.53 ms
64 bytes from 10.0.0.5: icmp_seq=6 ttl=64 time=5.38 ms
64 bytes from 10.0.0.5: icmp_seq=7 ttl=64 time=7.16 ms
64 bytes from 10.0.0.5: icmp_seq=8 ttl=64 time=7.69 ms
64 bytes from 10.0.0.5: icmp_seq=9 ttl=64 time=7.99 ms
64 bytes from 10.0.0.5: icmp_seq=10 ttl=64 time=8.79 ms
64 bytes from 10.0.0.5: icmp_seq=11 ttl=64 time=7.55 ms
64 bytes from 10.0.0.5: icmp_seq=12 ttl=64 time=8.28 ms
64 bytes from 10.0.0.5: icmp_seq=13 ttl=64 time=8.10 ms

mininet> h2 ping h5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
```

```
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=8.39 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=9.04 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=8.21 ms
64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=6.83 ms
64 bytes from 10.0.0.5: icmp_seq=5 ttl=64 time=9.73 ms
64 bytes from 10.0.0.5: icmp_seq=6 ttl=64 time=9.63 ms
64 bytes from 10.0.0.5: icmp_seq=7 ttl=64 time=6.68 ms
```

———————————————————————————————-

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=6.79 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=6.23 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=6.12 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=5.41 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=5.40 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=6.21 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=6.23 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=7.89 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=3.02 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=5.33 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=5.67 ms


mininet> h2 ping h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=7.62 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=6.48 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=5.93 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=1.87 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=7.52 ms
64 bytes from 10.0.0.3: icmp_seq=6 ttl=64 time=3.08 ms
64 bytes from 10.0.0.3: icmp_seq=7 ttl=64 time=7.14 ms
64 bytes from 10.0.0.3: icmp_seq=8 ttl=64 time=5.84 ms
64 bytes from 10.0.0.3: icmp_seq=9 ttl=64 time=6.56 ms
64 bytes from 10.0.0.3: icmp_seq=10 ttl=64 time=5.62 ms
64 bytes from 10.0.0.3: icmp_seq=11 ttl=64 time=3.40 ms


mininet> h3 ping h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=10.5 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=7.47 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=7.77 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=10.1 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=8.94 ms
64 bytes from 10.0.0.4: icmp_seq=6 ttl=64 time=9.33 ms
64 bytes from 10.0.0.4: icmp_seq=7 ttl=64 time=6.92 ms
64 bytes from 10.0.0.4: icmp_seq=8 ttl=64 time=9.44 ms
64 bytes from 10.0.0.4: icmp_seq=9 ttl=64 time=9.41 ms
64 bytes from 10.0.0.4: icmp_seq=10 ttl=64 time=4.19 ms
64 bytes from 10.0.0.4: icmp_seq=11 ttl=64 time=8.98 ms
```

```
64 bytes from 10.0.0.4: icmp_seq=12 ttl=64 time=8.20 ms
64 bytes from 10.0.0.4: icmp_seq=13 ttl=64 time=10.7 ms
```

Learning hub Data

THREE PINGS BETWEEN H2 AND H5

```
mininet> h2 ping h5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=22.6 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=0.312 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=0.097 ms
64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=0.115 ms
64 bytes from 10.0.0.5: icmp_seq=5 ttl=64 time=0.117 ms
64 bytes from 10.0.0.5: icmp_seq=6 ttl=64 time=0.109 ms
64 bytes from 10.0.0.5: icmp_seq=7 ttl=64 time=0.112 ms

mininet> h2 ping h5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=0.553 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=0.115 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=0.078 ms
64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=0.081 ms
64 bytes from 10.0.0.5: icmp_seq=5 ttl=64 time=0.086 ms
64 bytes from 10.0.0.5: icmp_seq=6 ttl=64 time=0.165 ms
64 bytes from 10.0.0.5: icmp_seq=7 ttl=64 time=0.085 ms
64 bytes from 10.0.0.5: icmp_seq=8 ttl=64 time=0.187 ms


mininet> h2 ping h5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=0.628 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=0.129 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=0.118 ms
64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=0.116 ms
64 bytes from 10.0.0.5: icmp_seq=5 ttl=64 time=0.087 ms
64 bytes from 10.0.0.5: icmp_seq=6 ttl=64 time=0.113 ms
64 bytes from 10.0.0.5: icmp_seq=7 ttl=64 time=0.116 ms
```

——————————————————————————————

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=6.47 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.591 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.109 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.110 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.110 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.109 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.099 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.106 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.111 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.061 ms
```

```
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.106 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.111 ms

mininet> h2 ping h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=8.71 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.197 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.105 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.100 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.106 ms
64 bytes from 10.0.0.3: icmp_seq=6 ttl=64 time=0.068 ms
64 bytes from 10.0.0.3: icmp_seq=7 ttl=64 time=0.116 ms
64 bytes from 10.0.0.3: icmp_seq=8 ttl=64 time=0.102 ms
64 bytes from 10.0.0.3: icmp_seq=9 ttl=64 time=0.106 ms
64 bytes from 10.0.0.3: icmp_seq=10 ttl=64 time=0.110 ms
64 bytes from 10.0.0.3: icmp_seq=11 ttl=64 time=0.102 ms
64 bytes from 10.0.0.3: icmp_seq=12 ttl=64 time=0.094 ms
64 bytes from 10.0.0.3: icmp_seq=13 ttl=64 time=0.107 ms


mininet> h3 ping h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=12.3 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=0.395 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.114 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.116 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=0.115 ms
64 bytes from 10.0.0.4: icmp_seq=6 ttl=64 time=0.113 ms
64 bytes from 10.0.0.4: icmp_seq=7 ttl=64 time=0.106 ms
64 bytes from 10.0.0.4: icmp_seq=8 ttl=64 time=0.067 ms
64 bytes from 10.0.0.4: icmp_seq=9 ttl=64 time=0.142 ms
64 bytes from 10.0.0.4: icmp_seq=10 ttl=64 time=0.056 ms
64 bytes from 10.0.0.4: icmp_seq=11 ttl=64 time=0.104 ms
64 bytes from 10.0.0.4: icmp_seq=12 ttl=64 time=0.231 ms
64 bytes from 10.0.0.4: icmp_seq=13 ttl=64 time=0.113 ms
```

Answer

In learning switch it detects the ports of destination and source and its Mac address and keeps it in the Mac address table so while forwarding the data after first time it only sends to the correct destination Mac while in Hub it does not learn the port of the host, it just floods the packet so latency is high and remains the same while in learning switch for the first time it takes more time but after that it takes less time as we can observe in the above results.

2) Run a throughput test between h1 and h5. Report the observed values. Explain the differences between the Hub Controller and Learning Switch.

Controller Hub Data(Bandwidth)

```
mininet> iperf h1 h5
*** Iperf: testing TCP bandwidth between h1 and h5
*** Results: ['13.9 Mbits/sec', '16.4 Mbits/sec']
mininet>
```

Learning hub Data(Bandwidth)

```
mininet> iperf h1 h5
*** Iperf: testing TCP bandwidth between h1 and h5
*** Results: ['27.3 Gbits/sec', '27.4 Gbits/sec']
```

Answer

- **Hub Controller**: The Hub Controller forwards all incoming traffic to all ports, leading to a lot of broadcast traffic, especially in cases where there are multiple hosts in the network. This unnecessary traffic can saturate the network and reduce the effective throughput between hosts h1 and h5.
- **Learning Switch**: The Learning Switch is more efficient in handling traffic. It learns MAC-to-Port mappings and forwards traffic only to the specific port where the destination host (e.g., h5) is connected. This reduces broadcast traffic and improves overall network efficiency, resulting in higher throughput between h1 and h5.

3) Run pingall in both cases and report the installed rules on switches.

Controller Hub Data

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5
h2 -> h1 h3 h4 h5
h3 -> h1 h2 h4 h5
h4 -> h1 h2 h3 h5
h5 -> h1 h2 h3 h4
*** Results: 0% dropped (20/20 received)
mininet>
```

Installed rules

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=681.389s, table=0, n_packets=8478, n_bytes=11582892, priority=0
actions=CONTROLLER:65535
*** s2 ------------------------------------------------------------------------
 cookie=0x0, duration=681.400s, table=0, n_packets=12185, n_bytes=11827554, priority=0
actions=CONTROLLER:65535
mininet>
```

-

Learning hub Data

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5
h2 -> h1 h3 h4 h5
h3 -> h1 h2 h4 h5
h4 -> h1 h2 h3 h5
h5 -> h1 h2 h3 h4
*** Results: 0% dropped (20/20 received)
```

Installed rules

```
mininet> dpctl dump-flows
*** s1 -----------------------------------------------------------------------
 cookie=0x0, duration=287.151s, table=0, n_packets=17, n_bytes=1498,
priority=1,in_port="s1-eth2",dl_dst=00:00:00:00:00:01 actions=output:"s1-eth1"
 cookie=0x0, duration=287.150s, table=0, n_packets=16, n_bytes=1400,
priority=1,in_port="s1-eth1",dl_dst=00:00:00:00:00:02 actions=output:"s1-eth2"
 cookie=0x0, duration=250.829s, table=0, n_packets=18, n_bytes=1596,
priority=1,in_port="s1-eth3",dl_dst=00:00:00:00:00:02 actions=output:"s1-eth2"
 cookie=0x0, duration=250.824s, table=0, n_packets=17, n_bytes=1498,
priority=1,in_port="s1-eth2",dl_dst=00:00:00:00:00:03 actions=output:"s1-eth3"
 cookie=0x0, duration=222.973s, table=0, n_packets=22, n_bytes=1876,
priority=1,in_port="s1-eth4",dl_dst=00:00:00:00:00:03 actions=output:"s1-eth3"
 cookie=0x0, duration=222.971s, table=0, n_packets=17, n_bytes=1498,
priority=1,in_port="s1-eth3",dl_dst=00:00:00:00:00:04 actions=output:"s1-eth4"
 cookie=0x0, duration=150.831s, table=0, n_packets=316028, n_bytes=20857884,
priority=1,in_port="s1-eth4",dl_dst=00:00:00:00:00:01 actions=output:"s1-eth1"
 cookie=0x0, duration=150.826s, table=0, n_packets=389536, n_bytes=17151052608,
priority=1,in_port="s1-eth1",dl_dst=00:00:00:00:00:05 actions=output:"s1-eth4"
 cookie=0x0, duration=33.410s, table=0, n_packets=3, n_bytes=238,
priority=1,in_port="s1-eth3",dl_dst=00:00:00:00:00:01 actions=output:"s1-eth1"
 cookie=0x0, duration=33.408s, table=0, n_packets=2, n_bytes=140,
priority=1,in_port="s1-eth1",dl_dst=00:00:00:00:00:03 actions=output:"s1-eth3"
 cookie=0x0, duration=33.395s, table=0, n_packets=2, n_bytes=140,
priority=1,in_port="s1-eth1",dl_dst=00:00:00:00:00:04 actions=output:"s1-eth4"
 cookie=0x0, duration=33.374s, table=0, n_packets=7, n_bytes=518,
priority=1,in_port="s1-eth4",dl_dst=00:00:00:00:00:02 actions=output:"s1-eth2"
 cookie=0x0, duration=33.372s, table=0, n_packets=2, n_bytes=140,
priority=1,in_port="s1-eth2",dl_dst=00:00:00:00:00:04 actions=output:"s1-eth4"
 cookie=0x0, duration=33.364s, table=0, n_packets=2, n_bytes=140,
priority=1,in_port="s1-eth2",dl_dst=00:00:00:00:00:05 actions=output:"s1-eth4"
 cookie=0x0, duration=33.342s, table=0, n_packets=2, n_bytes=140,
priority=1,in_port="s1-eth3",dl_dst=00:00:00:00:00:05 actions=output:"s1-eth4"
 cookie=0x0, duration=298.769s, table=0, n_packets=104, n_bytes=9750, priority=0
actions=CONTROLLER:65535
*** s2 -----------------------------------------------------------------------
```

```
 cookie=0x0, duration=222.986s, table=0, n_packets=18, n_bytes=1596,
priority=1,in_port="s2-eth2",dl_dst=00:00:00:00:00:03 actions=output:"s2-eth1"
 cookie=0x0, duration=222.981s, table=0, n_packets=23, n_bytes=1974,
priority=1,in_port="s2-eth1",dl_dst=00:00:00:00:00:04 actions=output:"s2-eth2"
 cookie=0x0, duration=150.845s, table=0, n_packets=316024, n_bytes=20857604,
priority=1,in_port="s2-eth3",dl_dst=00:00:00:00:00:01 actions=output:"s2-eth1"
 cookie=0x0, duration=150.837s, table=0, n_packets=389542, n_bytes=17151053084,
priority=1,in_port="s2-eth1",dl_dst=00:00:00:00:00:05 actions=output:"s2-eth3"
 cookie=0x0, duration=33.407s, table=0, n_packets=3, n_bytes=238,
priority=1,in_port="s2-eth2",dl_dst=00:00:00:00:00:01 actions=output:"s2-eth1"
 cookie=0x0, duration=33.387s, table=0, n_packets=3, n_bytes=238,
priority=1,in_port="s2-eth2",dl_dst=00:00:00:00:00:02 actions=output:"s2-eth1"
 cookie=0x0, duration=33.377s, table=0, n_packets=3, n_bytes=238,
priority=1,in_port="s2-eth3",dl_dst=00:00:00:00:00:02 actions=output:"s2-eth1"
 cookie=0x0, duration=33.356s, table=0, n_packets=3, n_bytes=238,
priority=1,in_port="s2-eth3",dl_dst=00:00:00:00:00:03 actions=output:"s2-eth1"
 cookie=0x0, duration=33.336s, table=0, n_packets=3, n_bytes=238,
priority=1,in_port="s2-eth3",dl_dst=00:00:00:00:00:04 actions=output:"s2-eth2"
 cookie=0x0, duration=33.335s, table=0, n_packets=2, n_bytes=140,
priority=1,in_port="s2-eth2",dl_dst=00:00:00:00:00:05 actions=output:"s2-eth3"
 cookie=0x0, duration=298.778s, table=0, n_packets=98, n_bytes=9118, priority=0
actions=CONTROLLER:65535
```

PART 2 (FIREWALL IMPLEMENTATION ON LEARNING SWITCH)

1) Run pingall and report the results.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 X h5
h2 -> h1 h3 h4 X
h3 -> h1 h2 h4 X
h4 -> X h2 h3 h5
h5 -> h1 X X h4
*** Results: 30% dropped (14/20 received)
```

2)Report the installed rules on both switches. Can you think of ways to minimize the number of firewall rules on the switch?

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=94.121s, table=0, n_packets=3, n_bytes=238, priority=1,in_port="s1-
eth2",dl_dst=00:00:00:00:00:01 actions=output:"s1-eth1"
 cookie=0x0, duration=94.120s, table=0, n_packets=2, n_bytes=140, priority=1,in_port="s1-
eth1",dl_dst=00:00:00:00:00:02 actions=output:"s1-eth2"
 cookie=0x0, duration=94.111s, table=0, n_packets=3, n_bytes=238, priority=1,in_port="s1-
eth3",dl_dst=00:00:00:00:00:01 actions=output:"s1-eth1"
 cookie=0x0, duration=94.109s, table=0, n_packets=2, n_bytes=140, priority=1,in_port="s1-
eth1",dl_dst=00:00:00:00:00:03 actions=output:"s1-eth3"
```

```
 cookie=0x0, duration=94.097s, table=0, n_packets=7, n_bytes=462, priority=1,in_port="s1-
eth4",dl_dst=00:00:00:00:00:01 actions=output:"s1-eth1"
 cookie=0x0, duration=84.086s, table=0, n_packets=3, n_bytes=182, priority=1,in_port="s1-
eth1",dl_dst=00:00:00:00:00:05 actions=output:"s1-eth4"
 cookie=0x0, duration=84.073s, table=0, n_packets=3, n_bytes=238, priority=1,in_port="s1-
eth3",dl_dst=00:00:00:00:00:02 actions=output:"s1-eth2"
 cookie=0x0, duration=84.069s, table=0, n_packets=2, n_bytes=140, priority=1,in_port="s1-
eth2",dl_dst=00:00:00:00:00:03 actions=output:"s1-eth3"
 cookie=0x0, duration=84.061s, table=0, n_packets=6, n_bytes=420, priority=1,in_port="s1-
eth4",dl_dst=00:00:00:00:00:02 actions=output:"s1-eth2"
 cookie=0x0, duration=84.060s, table=0, n_packets=2, n_bytes=140, priority=1,in_port="s1-
eth2",dl_dst=00:00:00:00:00:04 actions=output:"s1-eth4"
 cookie=0x0, duration=74.032s, table=0, n_packets=6, n_bytes=420, priority=1,in_port="s1-
eth4",dl_dst=00:00:00:00:00:03 actions=output:"s1-eth3"
 cookie=0x0, duration=74.030s, table=0, n_packets=2, n_bytes=140, priority=1,in_port="s1-
eth3",dl_dst=00:00:00:00:00:04 actions=output:"s1-eth4"
 cookie=0x0, duration=58.858s, table=0, n_packets=0, n_bytes=0, priority=1,in_port="s1-
eth1",dl_dst=00:00:00:00:00:04 actions=output:"s1-eth4"
 cookie=0x0, duration=48.880s, table=0, n_packets=0, n_bytes=0, priority=1,in_port="s1-
eth2",dl_dst=00:00:00:00:00:05 actions=output:"s1-eth4"
 cookie=0x0, duration=38.894s, table=0, n_packets=0, n_bytes=0, priority=1,in_port="s1-
eth3",dl_dst=00:00:00:00:00:05 actions=output:"s1-eth4"
 cookie=0x0, duration=96.286s, table=0, n_packets=95, n_bytes=8804, priority=0
actions=CONTROLLER:65535
*** s2 ------------------------------------------------------------------------
 cookie=0x0, duration=94.111s, table=0, n_packets=2, n_bytes=140, priority=1,in_port="s2-
eth2",dl_dst=00:00:00:00:00:01 actions=output:"s2-eth1"
 cookie=0x0, duration=84.099s, table=0, n_packets=4, n_bytes=280, priority=1,in_port="s2-
eth3",dl_dst=00:00:00:00:00:01 actions=output:"s2-eth1"
 cookie=0x0, duration=84.095s, table=0, n_packets=5, n_bytes=266, priority=1,in_port="s2-
eth1",dl_dst=00:00:00:00:00:05 actions=output:"s2-eth3"
 cookie=0x0, duration=84.073s, table=0, n_packets=3, n_bytes=238, priority=1,in_port="s2-
eth2",dl_dst=00:00:00:00:00:02 actions=output:"s2-eth1"
 cookie=0x0, duration=84.070s, table=0, n_packets=6, n_bytes=420, priority=1,in_port="s2-
eth1",dl_dst=00:00:00:00:00:04 actions=output:"s2-eth2"
 cookie=0x0, duration=84.062s, table=0, n_packets=2, n_bytes=140, priority=1,in_port="s2-
eth3",dl_dst=00:00:00:00:00:02 actions=output:"s2-eth1"
 cookie=0x0, duration=74.047s, table=0, n_packets=3, n_bytes=238, priority=1,in_port="s2-
eth2",dl_dst=00:00:00:00:00:03 actions=output:"s2-eth1"
 cookie=0x0, duration=74.030s, table=0, n_packets=2, n_bytes=140, priority=1,in_port="s2-
eth3",dl_dst=00:00:00:00:00:03 actions=output:"s2-eth1"
 cookie=0x0, duration=54.007s, table=0, n_packets=3, n_bytes=238, priority=1,in_port="s2-
eth3",dl_dst=00:00:00:00:00:04 actions=output:"s2-eth2"
 cookie=0x0, duration=54.004s, table=0, n_packets=2, n_bytes=140, priority=1,in_port="s2-
eth2",dl_dst=00:00:00:00:00:05 actions=output:"s2-eth3"
 cookie=0x0, duration=96.297s, table=0, n_packets=85, n_bytes=7928, priority=0
actions=CONTROLLER:65535
```

Answer

- Use wildcards to match multiple criteria with a single rule.

- Implement a default-deny policy and create rules only for allowed traffic.
- Group rules with similar characteristics or objectives together.
- Prioritize rules based on importance or frequency.
- Utilize address and port ranges in rules.
- Consider variables and macros for rule definitions.
- Regularly review and clean up obsolete rules.
- Implement service-based rules instead of device-specific rules.
- Segment networks to control access at network boundaries.
- Consider advanced networking devices for more granular rule management.

3) Suppose the network operator intends to implement firewall policies in real time. How can she ensure that the pre-existing rules do not interfere with the firewall policy?

- Prioritize firewall rules with higher priorities.
- Use specific matching criteria for firewall rules.
- Add firewall rules before pre-existing rules in flow tables.
- Consider separate flow tables for firewall rules.
- Dynamically manage and remove firewall rules as needed.
- Implement logging and monitoring for quick detection of issues.
- Test and simulate firewall rules before deployment.
- Maintain clear and updated documentation of all rules.

PART 3 (LOAD BALANCER IMPLEMENTATION OF LEARNING SWITCH)

1) Have the three hosts (H1, H2, and H3) ping the virtual IP and report the installed rule on the switches.

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=2.872s, table=0, n_packets=34, n_bytes=3764, priority=0
actions=CONTROLLER:65535
*** s2 ------------------------------------------------------------------------
 cookie=0x0, duration=2.877s, table=0, n_packets=34, n_bytes=3764, priority=0
actions=CONTROLLER:65535


mininet> h1 ping h5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=9.03 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=0.282 ms
```

```
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=0.071 ms
64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=0.115 ms
--- 10.0.0.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3052ms
rtt min/avg/max/mdev = 0.071/2.375/9.032/3.844 ms
```

<u>Pinging Virtual IP</u>

```
mininet> h1 ping 10.0.0.42

------- ARP Packet is being handled ---------
We are Generating ARP Reply Packet
ARP request ip address of client : 10.0.0.1, client mac: 00:00:00:00:00:01
Selected server MAC: 00:00:00:00:00:04
Done after we have the ARP reply packet processed
the ARP reply packet has been sent
packet in 1 00:00:00:00:00:01 00:00:00:00:00:04 1
-----------------------------------
--------------- We are Handling TCP Packet -------------
<==== Added TCP Flow- Route to Server: 10.0.0.4 from Client :10.0.0.1 on Switch Port:4====>
<==== Added TCP Flow- Reverse route from Server: 10.0.0.4 to Client: 00:00:00:00:00:01 on
Switch Port:1====>
TCP packet handled: True
packet in 2 4a:d0:1a:ad:cc:10 33:33:00:00:00:fb 1
packet in 1 be:36:b8:14:15:50 33:33:00:00:00:fb 4
packet in 2 00:00:00:00:00:01 00:00:00:00:00:04 1
-----------------------------------
--------------- We are Handling TCP Packet -------------
TCP packet handled: False
packet in 2 00:00:00:00:00:04 ff:ff:ff:ff:ff:ff 2
packet in 1 00:00:00:00:00:04 ff:ff:ff:ff:ff:ff 4
packet in 2 00:00:00:00:00:04 00:00:00:00:00:01 2
-----------------------------------
```

2) If you were to implement a load balancing policy that considers the load on these servers, what additional steps would you take? [No need to implement it in the code, just describe the additional steps.]

- **Health Monitoring**: Implement a health monitoring mechanism to periodically check the status and load of each server in the load balancing pool. This can involve sending health check requests (e.g., HTTP GET requests) to the servers and analyzing their responses. If a server is deemed unhealthy or overloaded, it should be temporarily removed from the pool.
- **Dynamic Server Addition/Removal**: Implement logic to dynamically add or remove servers from the load balancing pool based on their health status and load. When a

server becomes unhealthy or overloaded, it should be temporarily removed from the pool to avoid routing traffic to it. Conversely, when a server recovers or becomes less loaded, it should be reintroduced into the pool.

- **Load Balancing Algorithms**: Implement load balancing algorithms that distribute incoming traffic among the healthy servers. Common load balancing algorithms include Round Robin, Least Connections, and Weighted Round Robin. These algorithms can be used to determine which server should handle each incoming request.
- **Load Balancer Statistics**: Keep track of statistics related to the server's load, such as the number of active connections, response times, and resource utilization. These statistics can be used to make informed load balancing decisions.
- **Session Persistence**: Implement session persistence mechanisms to ensure that multiple requests from the same client are directed to the same server. This is important for applications that rely on session state. Session persistence can be achieved through techniques like source IP affinity or using cookies.
- **Failover Handling**: Develop a failover strategy to handle cases where all servers in the pool are unhealthy or overloaded. In such situations, traffic can be directed to a backup server or an error page can be displayed to users.

———————————————— END OF REPORT ————————————————